Refine Search

Search Results -

Terms	Documents
L6 and L12	2

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search:

Refine Search

Recall Text Clear Interrupt

Search History

DATE: Thursday, February 02, 2006 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		
L25	l6 and l12	2	L25
L24	l13 and redirector	4	L24
L23	l10 and l20	0	L23
L22	l11 and l20	0	L22
L21	l15 and l20	0	L21
L20	5940594.pn.	2	L20
	DB=USPT; PLUR=YES; OP=OR		
L19	(5530855 | 4432057 | 5806074 | 5794253 | 5251316 | 5806075 | 5404505 | 5535386 | 5347653 | 4961139 | 5745899 | 5499367 | 5796999 | 5551027 | 5325524 | 5434994 | 5448727 | 5394394 | 5276871 | 5530851 | 5113519)![PN]	21	L19
L18	('5924096')[PN]	1	L18
L17	(6128627 | 6098064 | 5940594 | 6173311 | 6148394 | 5924096)![PN]	6	L17
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR		

L16	('6487641')[ABPN1,NRPN,PN,TBAN,WKU]	2	L16
L15	L14 and (database or data with base) near3 table	38	L15
L14	L13 and dataset$2	44	L14
L13	L12 and cache near5 databas$2	112	L13
L12	updat$4 near4 cache near8 serv$2 and quer$4 near9 cache	201	L12
L11	711/113	1902	L11
L10	711.clas.	28320	L10
L9	709/240	735	L9
L8	709.clas.	41892	L8
L7	707.clas.	32201	L7
L6	707/206	1188	L6
L5	707/200	4334	L5
L4	707/104.1	5381	L4
L3	707/100	7285	L3
L2	707/10	11287	L2
L1	707/1	7419	L1

END OF SEARCH HISTORY

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#
End of Result Set

Generate Collection Print

L24: Entry 4 of 4 File: USPT Nov 17, 1998

US-PAT-NO: 5838916
DOCUMENT-IDENTIFIER: US 5838916 A

TITLE: Systems and methods for executing application programs from a memory device linked to a server

DATE-ISSUED: November 17, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Domenikos; Steven D.	Millis	MA	02054	
Domenikos; George C.	Chelsea	MA	02150	

APPL-NO: 08/818665 [PALM]
DATE FILED: March 14, 1997

PARENT-CASE:
This application is a continuation-in-part of U.S. Ser. No. 08/616,746 filed 14 Mar. 1996 and entitled "Systems and Methods for Executing Application Programs from a Memory Device Linked to a Server at an Internet Site," the teachings of which are incorporated herein by reference.

INT-CL-ISSUED: [06] G01 D 1/16

US-CL-ISSUED: 395/200.49; 395/200.47, 345/330
US-CL-CURRENT: 709/219; 709/217, 715/753

FIELD-OF-CLASSIFICATION-SEARCH: 395/200.49, 395/200.47, 395/200.8, 395/838, 395/450, 395/467, 345/330, 345/335, 707/10, 707/1, 707/104
See application file for complete search history.

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4825354	April 1989	Agrawal et al.	707/10

OTHER PUBLICATIONS

Felton et al., "Web Spoofing: An Internet Con Game", Technical Report 540-96 (revised Feb. 1997) Department of Computer Science, Princeton University, pp. 1-9.

Andreesen et al., "Netscape Enterprise Vision and Product Roadmap", Netscape Products, Apr. 23, 1997, pp. 1-40. URL: home.netscape.com/newsref/std/index.html#white--papers.
Sharpe, "Just What is SMB", Copyright .COPYRGT.1996, Richard Sharpe, Dec. 1996, pp. 1-9.
Bach, "5.14 -Mounting and Unmounting File Systems", The Design of the Unix Operating system, pp. 119-141.
Callaghan, "WebNFS.TM. The Filesystem for the World Wide Web", Sunsoft, Inc. May 1996, pp. 1-20. URL:www.sun.com/solaris/networking/webnfs/webnfs.html.

ART-UNIT: 274

PRIMARY-EXAMINER: Peeso; Thomas

ATTY-AGENT-FIRM: Kelly; Edward J. Foley, Hoag & Eliot LLP

ABSTRACT:

Systems and processes of the invention allow a computer to connect to a server of an Internet site for executing an application program that is stored on a disk linked to that server. Specifically, processes are disclosed that provide a data transport interface for connecting to the server. The processes transport from the server a server address signal representative of a network address of the server, and a path name signal representative of a file system that includes an application program. The process further generates a mount request as a function of the path name signal and transports the mount request to the server to direct the server to provide an array of server file pointers that point to a file descriptor representative of the file system that includes the application program. The processes generate an array of remote file pointers, as a function of the server address signal and the array of server file pointers, and provide the array of remote file pointers to a program loader that transports from the disk at the server site to a local program memory element, a file block associated with the selected remote file pointer. The process can employ a redirector that translates requests to access remote files into HTTP compliant commands for collecting files from an HTTP server site.

17 Claims, 9 Drawing figures

Previous Doc Next Doc Go to Doc#

L24: Entry 4 of 4 File: USPT Nov 17, 1998

DOCUMENT-IDENTIFIER: US 5838916 A
TITLE: Systems and methods for executing application programs from a memory device linked to a server

Abstract Text (1):
Systems and processes of the invention allow a computer to connect to a server of an Internet site for executing an application program that is stored on a disk linked to that server. Specifically, processes are disclosed that provide a data transport interface for connecting to the server. The processes transport from the server a server address signal representative of a network address of the server, and a path name signal representative of a file system that includes an application program. The process further generates a mount request as a function of the path name signal and transports the mount request to the server to direct the server to provide an array of server file pointers that point to a file descriptor representative of the file system that includes the application program. The processes generate an array of remote file pointers, as a function of the server address signal and the array of server file pointers, and provide the array of remote file pointers to a program loader that transports from the disk at the server site to a local program memory element, a file block associated with the selected remote file pointer. The process can employ a redirector that translates requests to access remote files into HTTP compliant commands for collecting files from an HTTP server site.

Brief Summary Text (18):
The methods of the invention can include methods for allowing a client to execute an application program stored on a memory device linked to an HTTP server, and can comprise the steps of mounting a remote file system containing the application program and being linked to the HTTP server, monitoring the operating system to detect file system requests for files stored within the remote file system and passing the file system requests to a redirector element for translating the file system requests into HTTP compliant signals for transmission across a network. In this practice of the invention, the step of mounting can include the step of generating an array of file pointers representative of pointers to files associated with the application program.

Brief Summary Text (19):
In a further practice, the processes of the invention can include the step of providing plural redirector elements for translating the file system requests into a selected one of plural transport protocols. For example, the client element can have a network interface to a plurality of different networks each of which use a different file transport protocol, such as NCP, NFS, SMB or any other of the open or proprietary services that provide protocol for transmitting or sharing data, such as files, across network media. In this practice, each one of the redirector elements can be adapted for employment with one or more of the data transport protocols, thereby allowing the client element to access any one of a plurality of servers each having a different remote file system. The process can include a step of selecting one of the plural redirector elements to translate the file system requests. The step of selecting the redirector element can be made as a function of

any suitable criteria or characteristic including the types of file systems, as well as a measure of the transmission latency time of each of the networks, as well as the transmission protocols of the networks. In one practice of the invention, the plural redirectors are provided with a wrapper interface that, in a way, encapsulates each of the redirector elements such that all the redirector elements are provided with the same interface to the operating system of the client computer. This is understood to provide portability and to reduce the complexity of the overall system.

Brief Summary Text (20):
The term "redirector" as used herein will be understood to encompass a program or process that can direct a request for data or services to a selected source of data or services and optionally can translate file system requests into commands or data signals that comply with at least one network protocol for transmission of data across network media.

Brief Summary Text (21):
In a further embodiment, the processes of the invention include the further steps of providing the client with access to a cache memory for cache storing portions of the file system which contain the application programs, and a further step of providing a cache redirector element that responds to client file requests by accessing portions of the file system stored within the cache memory. The accessed portions of the file system can be employed for running the application program within the clients local program memory. In one practice, the process can include the step of directing the cache memory to mirror the file system being stored therein. Accordingly, in this practice, a mirroring process can operate such that as changes are detected within the file system being exported by the server, the file, and file or directory structure maintained within the cache memory device, are similarly updated. Optionally, this mirroring can be done automatically, or upon the instruction of the user.

Brief Summary Text (31):
In a further embodiment, the invention can be understood as server processes that allow a client to execute locally an application program stored at a memory device of a server. These processes can include the steps of providing an HTTP server process that can access a memory device having storage for one or more server directories that store files associated with an application program, including the executable code of the application program. The server provides the client with access to the files and directories, and the client can cache portions of the file system containing the application program, and employ a cache redirector element to access the portions of the file system that is stored within the cache memory.

Detailed Description Text (2):
The invention comprises, inter alia, systems and methods that enable a web site administrator to provide links to remote applications within their web pages. One realization of the invention is that a remote client can be allowed to execute an application program stored at a HTTP server. To this end, the invention provides the server with access to a remote file system, and with access to a file that can contain information of the files, directory structure and system environment associated with the application program. The invention can also include a system for allowing the client to mount the portion of the shared file system that includes the application program. The system can include an HTTP redirector element that translates operating system requests to access portions of the shared file system into HTTP compliant commands. An HTTP server at an Internet site responds to the commands and provides access to the HTTP server directories. Accordingly, the HTTP redirector creates a drive connection to the server directories to provide a client with access to files stored therein. This provides an HTTP shared file system. In this way, a remote application can be mounted into the client file system and then run from within a window of a browser program or from a launching program. Optionally, the links can appear as program icons within the window of the

browser or launcher program.

Detailed Description Text (11):
FIG. 2 depicts one process for providing a web page application link. In this process, the application program is provided within a shared directory, or an exported file system. The directories are to be shared through a file system server such as a LAN manager system, the NFS system, the NCP system or any other suitable file system server, or optionally by employing the HTTP redirector, can be placed on an HTTP server. For the application program in the shared directory, the web site administrator can, as shown by step 21, generate an application information file for the remote application. Optionally, the file can be created automatically by a process executing on the server.

Detailed Description Text (43):
FIG. 6 figuratively depicts the components and interaction of an alternative system according to the invention that includes a cache memory system, an HTTP redirector and a cache redirector. In particular, FIG. 6 depicts a system that includes a client element 212, am HTTP server 214, a cache system 216, a file system server 218, a browser 222, a browser interface 224, a launcher program 226, a file system connection process 228, a remote access connection 230, a networking API 234, a redirector API 236, a file system cache API 238, an operating system file system interface 240, a cache redirector 242, a network file system client 250, an optional NFS redirector 252, an HTTP redirector 254, a further redirector 258, and a mini-redirector kernel element 260.

Detailed Description Text (47):
FIG. 7 depicts, that in addition to running remote applications from the network through web pages presented by a server 214, applications can also be run from the cache memory 216, optionally by activating links in an Off-line page. The Off-line page can be an HTML page stored locally on the client 212, and can have links to applications stored in the cache memory 216, for launching these cached programs. Moreover, the client can also include a process for automatically launching applications from the cache memory. Typically, this process is included in the cache redirector 242. This process can determine if the same application is available from both the server and the cache. If accessing the cache is more efficient, the process will launch the cached version of the application.

Detailed Description Text (48):
As described above, the operating system handles the request to load the remote application and routes file system requests to access remote application files to the appropriate network file system redirector. This can also include routing the requests to the cache redirector 242. The cache redirector 242 intercepts certain file system requests for remote files stored on the created virtual drive associated with the remote application. The cache redirector can determine whether the requested file has been cached. If the requested file has been cached the cache redirector 242 compares file attributes of the file in the local cache memory 216 with the attributes of the remote file. If the attributes indicate the remote files have been changed, the cache redirector deletes the cached copy.

Detailed Description Text (49):
If the requested file cannot be taken from the cache memory 216, the cache redirector 242 passes the request to one of underlying network file system clients. The cache redirector 242 then finds a space in the cache memory to store that file and marks it to be cached when the OS file system reads the file from the shared network file system. After collecting the file, the cache redirector 242 returns the file to the OS file system in response to the file system request.

Detailed Description Text (50):
After the application terminates, the launcher process 226 performs cleanup and disconnects the drive/universal naming convention (UNC) connection, restores the

environment to the original state, queries the cache redirector 242 about all remote applications which have components that have been fully cached. Updates the Off-line page, copying off-line application information files to an install directory, and adds a link for the any new entry into the Off-line page and/or removes any links that a file bumped out of the cache 216. The client 212 notifies the server 214 of the application termination event, or of execution failure event. The server 214 can then perform any necessary metering management required for the application.

Detailed Description Text (51):
In addition to the information described above, the application information file can contain information for the Off-line Page. This can include an Off-line allowance flag. If this flag is not set, the launcher 226 will not attempt to add the remote application represented by this application information file to the Off-line Page. This flag can be set by the server administrator program when the web administrator creates or updates the application information file. This information can also include the list of the components of the remote application that are to be cached in the cache 216 to run the application off-line. This list can also be set by the server administration program when the web administrator creates the application information file. Optionally, the list can include an Off-line preload request flag. This flag is set to indicate to the launcher 226 that the user requested to add the remote application represented by the application information file to the Off-line Page. The launcher 226 can include a process that will preload all of the components of the remote application to the cache memory 216 and update the Off-line Page, but not launch the remote application. The preload process can operate by generating the proper file requests for the selected application programs, and sending the file requests to the HTTP redirector.

Detailed Description Text (54):
Returning to FIG. 6, it can be seen that the cache memory system 216 can act as a database on the client's local disk which caches, or mirrors, remote applications and associated files when an application is run via clicking on an application link. The database can be located in a hidden directory under a client installation process directory. Files can be stored in the cache in a different format than the directory structure on the server, and therefore cannot be interpreted by a local user directly. The cache 216 can improve performance by avoiding the need to run remote applications over slow links. On subsequent requests to execute the same link, applications and associated files can be read from the cache rather than over the network. Additionally the cache can allow completely cached applications to be run from the Off-line page when there is no network connection to a remote file server containing the applications.

Detailed Description Text (55):
The depicted cache redirector 242 lies between the operating system's central file system control, and the installed network file systems. It connects the different file system services, NFS, HTTP, etc., to implement the file caching mechanism. All file system requests sent to this layer are routed to the appropriate network file system.

Detailed Description Text (56):
To this end, the network file system client 250 that can act as a multiple-protocol file system that takes remote I/O requests from the client and sends them to the appropriate server for processing. In the depicted embodiment, the network file system client 250 includes a plurality of redirectors that can act as a file system that takes remote I/O requests, for files, printers, serial ports, named pipes, mail slots, or other devices or abstractions, and sends the request to a server. As shown, each redirector in the network file system 250 can redirect I/O requests for use with a selected type of protocol for sharing files, data or devices.

Detailed Description Text (57):

The network file system client 250 can include a controller process that selects one of the plural redirectors for servicing a remote I/O request. The controller can select the redirector by determining from the I/O request, the server that has access to the requested data or device. Additionally, the controller can include a list that identifies application programs, or other data or devices, that are available from the multiple servers. The controller can select one of servers to access based on a characteristic or criteria such as the file systems or transport protocols available. For example, the NFS redirector may always be selected over the HTTP redirector, if an application program is available from both types of servers. Alternatively, the controller process can determine the transmission latency period, which defines the current response time of the network to a client request, to select the redirector of the faster network. The controller process can be part of a wrapper that encapsulates the plural redirectors and acts as an interface to the client operating system.

Detailed Description Text (58):
The NFS redirector 252 can be a file system client which handles requests to connect to exported remote file systems and to run remote applications shared by NFS servers. The NFS redirector 252 can be implemented as a V.times.D in Windows 95 and as an FSD in Windows NT. If Windows 3.1 and Windows for Workgroup are supported, then the NFS redirector 252 can be implemented as a redirector V.times.D. The core part of the redirector that provides NFS protocol, the protocol specific caching and the other features (that pertain to either the protocol or to the client redirector specific functions) common for all OS platforms can be implemented as a separate, OS independent, system component that is portable across the platforms. This core part is depicted in FIG. 6 as the mini-redirector 260. The shell of the depicted NFS redirector 252 is configured to support the client operating system and can be implemented as a network file system wrapper that is specific to client particular OS.

Detailed Description Text (59):
The depicted HTTP redirector 254 can act as a network file system client redirector which abstracts full URL pathnames, which are provided within the application information file. This provides an HTTP shared file system. The HTTP redirector 254 can then provide access to files within the HTTP shared file system by maintaining a Universal Naming Convention (UNC)/drive connection to the HTTP server directories. From the point of view of a remote application running on a local machine, the results of accessing files through the HTTP redirector 254 are the same as the results of accessing files through any other known network file system (i.e. Microsoft Network, or NFS). The HTTP redirector 254 can be a computer program implemented in the C++computer language. The program can monitor or respond to file requests made by the client operating system. The program will process those file requests for files stored at an HTTP server. In one embodiment, the program translates the file system requests into HTTP compliant command signals, which can be processed by an HTTP server. The HTTP compliant command signals can include simple requests or full requests, including the GET command, or any other HTTP commands or methods suitable for communicating with an HTTP server. The server responds to the commands by transferring, according to the HTTP protocol, the files identified by the file handles generated from URL pathnames. This provides an HTTP shared file system that directs the HTTP server to act as a network device. The development of the HTTP redirector 254 follows from well known principles in the art of software engineering and the actual coding of such a redirector is within the skill of one of ordinary skill in the art of computer programming.

Detailed Description Text (60):
The HTTP redirector 254 works in tight integration with the cache memory 216. In alternative embodiments, the HTTP redirector 254 may provide a process for reading a portion of a file rather than requiring the entire file to be read at once. The HTTP redirector 254 will provide the read access to the remote application files. The limited write access can be provided, for example, by employing the HTTP PUT

command.

Detailed Description Text (61):
It will be seen that an advantage of the HTT? redirector 254 is that the redirector 254 provides support for running application links over the Internet and through a firewall. Utilizing well known file system client and server software is sufficient for running application links over the Intranet, however allowing access to one of these well known file system servers over the Internet creates a security hole. Alternatively, an HTTP server can be placed on the network's firewall. All applications desired to be accessible from the Internet are to reside on that web server, and application links can be set up accordingly. The applications can then be run over the Internet via the HTTP redirector 254.

Detailed Description Text (62):
The HTTP redirector 254 can be implemented similarly to the NFS redirector, with a core portion implemented within a miniredirector and a wrapper for interfacing with the client OS.

CLAIMS:

1. Method for allowing a client to execute an application program stored on a memory device linked to an HTTP server, comprising the steps of

mounting a remote file system containing the application program and being linked to the HTTP server,

monitoring the operating system to detect file system requests for files stored within said remote file system,

providing plural redirector elements for translating said file system requests into selected transport protocols, wherein at least one of said redirector elements includes an HTTP redirector for translating the file system requests into HTTP complaint signals, and

passing the file system requests to at least one of said redirector elements for translating the file system requests into signals for transmission across a network.

2. A method according to claim 1, wherein said step of providing plural redirector elements includes the step of providing an NFS redirector element for translating requests into NFS compliant signals.

3. A method according to claim 1, wherein said step of providing plural redirector elements includes the step of providing a SMB redirector element for translating requests into SMB compliant signals.

4. A method according to claim 1, including the further step of, selecting one of said plural redirector elements to translate said file system requests.

5. A method according to claim 4, including the step of selecting said redirector as a function of said file system.

6. A method according to claim 4, wherein said client couples to a plurality of networks and including the further step of selecting said redirector as a function of comparing the transmission latency time characteristic for each of said plural networks.

7. method according to claim 4, wherein said client couples to a plurality of networks and including the further step of selecting said redirector as a function of the transmission protocols of said plural networks.

8. A method according to claim 1, including the step of providing said plural redirector elements with a wrapper for interfacing said redirectors with said operating system.

10. Method for allowing a client to execute an application program stored on a memory device linked to an HTTP server, comprising the steps of

providing said client with access to a cache memory for cache storing portions of said remote file system,

providing a cache redirector element for directing said client to access portions of said file system stored within said cache memory,

providing an HTTP redirector element for translating the file system requests into HTTP compliant signals,

mounting a remote file system containing the application program,

monitoring the operating system to detect file system requests for files stored within said remote file system, and

passing the file system requests to at least one of said redirector elements for accessing portions of said file system.

15. A method of allowing a client to install an application program stored on a memory device linked to a server, comprising the steps of

providing said server with access to an installation program for installing said application program for operation by said server,

generating an application information file representative of files associated with said installation program,

detecting a request from said client to install said application program,

transmitting to said client, responsive to said application information file, data representative of said files associated with said installation program,

providing said client with access to a cache memory for cache storing portions of said files associated with said installation program, and

providing a cache redirector for allowing said client to access portions of said files within said cache memory.

17. A method for allowing a server to provide remote execution of an application program stored on a memory device linked to said server, comprising the steps of

generating an application information file representative of files associated with said application program,

generating a link representative of a pointer to said application information file,

detecting a request from a client to execute said application program,

transmitting, responsive to said request, data signals to said client representative of said files associated with said application program,

providing said client with access to a cache memory for cache storing portions of

said file associated with said application program, and

providing a cache redirector element for allowing said client to access said portions of said file system stored within said cache memory.

Previous Doc Next Doc Go to Doc#